

On the Complexity of Restoring Corrupted Colorings*

Marzio De Biasi[†]

Juho Lauri[‡]

January 10, 2017

Abstract

In the r -FIX problem, we are given a graph G , a (non-proper) vertex-coloring $c : V(G) \rightarrow [r]$, and a positive integer k . The goal is to decide whether a proper r -coloring c' is obtainable from c by recoloring at most k vertices of G . Recently, Junosza-Szaniawski, Liedloff, and Rzażewski [SOF-SEM 2015] asked whether the problem has a polynomial kernel parameterized by the number of recolorings k . In a full version of the manuscript, the authors together with Garnero and Montealegre, answered the question in the negative: for every $r \geq 3$, the problem r -FIX does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP/poly}$. Independently of their work, we give an alternative proof of the theorem. Furthermore, we study the complexity of r -SWAP, where the only difference from r -FIX is that instead of k recolorings we have a budget of k color swaps. We show that for every $r \geq 3$, the problem r -SWAP is $\text{W}[1]$ -hard whereas r -FIX is known to be FPT. Moreover, when r is part of the input, we observe both FIX and SWAP are $\text{W}[1]$ -hard parameterized by treewidth. We also study promise variants of the problems, where we are guaranteed that a proper r -coloring c' is indeed obtainable from c by some finite number of swaps. For instance, we prove that for $r = 3$, the problems r -FIX-PROMISE and r -SWAP-PROMISE are NP-hard for planar graphs. As a consequence of our reduction, the problems cannot be solved in $2^{o(\sqrt{n})}$ time unless the Exponential Time Hypothesis (ETH) fails.

1 Introduction

Computational models are sometimes too idealized, and do not capture all information available or relevant to a problem. Moreover, in a dynamic world, constraints change over time due to more information becoming available. A problem arising frequently in practice in e.g., scheduling [27] is graph coloring. An assignment of r colors to the vertices of a graph $G = (V, E)$ is an r -coloring. Formally, an r -coloring $c : V \rightarrow [r]$ is said to be *proper* if $c(u) \neq c(v)$ for every $uv \in E$. In the graph coloring problem, the goal is to find the smallest r for which a graph is r -colorable. This quantity is known as the *chromatic number* of G , and is denoted by $\chi(G)$. Graph coloring is one of the most central problems in discrete mathematics and optimization. For a general introduction to the topic, we refer the reader to the book [17].

Suppose we have a graph G for which we have computed a proper vertex-coloring using $\chi(G)$ colors. Then due to constraints changing, a malicious adversary, or e.g., a system failure, the coloring is corrupted by redistributing the vertex colors over the graph G . How hard is it to restore the corrupted coloring to some optimal proper coloring of G ? In 2006, Clark, Holliday, Johnson, Trimm, Rubalcaba, and Walsh [5] introduced and investigated this problem under the name *chromatic villainy*. For restoring a corrupted coloring, they used the following operation called a *swap*. A swap between two distinct vertices u and v is the operation of assigning to u the color that appears on v , and vice versa. Formally, let c be a vertex-coloring of a graph G . The *villainy* of c , denoted by $B(c)$, is the minimum number of swaps to be performed to transform c into some proper vertex-coloring of G . The quantity $B(c)$ can also be seen as the minimum number of recolorings with the constraint that each color used in the new coloring c' must be used the same number of times as in c . In addition to the graph-theoretic viewpoint, there has been interest in the computational aspects of chromatic villainy [31].

Very recently, Junosza-Szaniawski, Liedloff, and Rzażewski [20] studied a problem they call r -FIX. In the problem, we are given a graph $G = (V, E)$, a non-proper r -coloring c of $V(G)$, and an integer k .

*Work partially supported by the Emil Aaltonen Foundation (J.L.).

[†]E-mail: marziodebiasi@gmail.com

[‡]Tampere University of Technology, Finland. E-mail: juho.lauri@tut.fi

The task is to decide whether a proper vertex-coloring c' is obtainable from c using at most k vertex recolorings. The authors observe the problem is NP-complete, and give several positive complexity results as well. In particular, using the framework of Björklund, Husfeldt, and Koivisto [1] they obtain a $O^*(2^n)$ -time and exponential space algorithm, where n is the number of vertices in the input graph. Furthermore, they show that for any fixed r , the problem is fixed-parameter tractable (FPT) parameterized by the number of recolorings k . Finally, in the same paper, the authors show that for graphs of treewidth t , the problem can be solved in $O(nr^{t+2})$ time. For a discussion on several related reoptimization and reconfiguration problems, we refer the reader to [20]. We also note that their results are considerably expanded in a full version of the manuscript currently under review [13].

The main difference between r -FIX and chromatic villainy as defined by Clark *et al.* [5] is the basic operation used. In r -FIX it is a recoloring, whereas in chromatic villainy it is a swap. For this reason, we shall refer to the computational problem arising from chromatic villainy as r -SWAP. That is, the input to r -SWAP is exactly the same as it is in r -FIX, but instead of k recolorings we have a budget of k swaps. Strictly speaking, there is another difference. In fact, in chromatic villainy, the corrupted vertex-coloring is not an arbitrary one, but has an additional property that could possibly be exploited. Namely, the property is that by using some finite number swaps, a proper vertex-coloring is indeed obtainable from the given one. Additional promises of properties of a problem are captured by the notion of a *promise problem*. Promise problems were introduced and studied by Even, Selman, and Yacobi [9], and they have several applications (for a survey, see [14]). In fact, it seems fair to argue that promise problems model real-world problems more accurately. Indeed, Goldreich [14] writes: “... I contend that almost all readers refer to this notion when thinking about computational problems, although they may be often unaware of this fact”. Motivated by these facts, we also separate two additional problems, r -FIX-PROMISE and r -SWAP-PROMISE, for which the input is guaranteed to satisfy additional properties (a precise definition is provided in Section 2).

In this context, it is natural to investigate whether a hard problem becomes easy when the set of instances is restricted. A priori, it is unknown how the addition of a promise affects the computational complexity of a problem. For example, it is hard to decide whether a graph contains a Hamiltonian cycle, even if we are promised it contains at most one such cycle [18]. In the other direction, one can efficiently find a satisfying assignment for an n -variable SAT formula that promises to have at least $2^n/n$ satisfying assignments. However, as shown by Valiant and Vazirani [30], it is hard to find an assignment even when the formula is promised to have exactly one solution.

Motivation. The problems r -FIX and r -SWAP are tightly related to local search which is a core technique in solving combinatorial optimization and operations research problems in practice. In local search, one aims to improve upon the current solution by replacing it with a better solution in its neighborhood. Specifically, the neighborhood is defined by the set of allowed operations that modify the current solution. Plausibly, the larger the neighborhood, the less likely is the local search to get stuck in a local optimum. On the other hand, the allowed operations should not be too demanding to compute. In fact, there has been significant interest in applying methods from parameterized complexity to analyzing local search procedures (see e.g., [7, 11, 21, 23, 29]).

The studied problems are also related to combinatorial reconfiguration problems, in particular *r -coloring reconfiguration* [4, 19, 32]. In this problem, we are given two proper r -colorings for a graph, and asked whether one can be transformed into the other by changing one color at a time, maintaining a proper coloring throughout. We argue that in the context of local search, the property of maintaining a proper r -coloring at each step can be relaxed: we are only interested in eventually arriving at a solution. To further motivate the use of promise conditions, we remark that there are r -coloring problems in which we know the sizes of the color classes (if an r -coloring exists). These include well-known problems arising from coding theory, such as partitioning of the n -dimensional Hamming space into binary codes with certain properties [28].

Our results. We continue the investigation of the complexity of restoring corrupted colorings. Specifically, we further study the complexity of r -FIX, under different basic operations and/or promise conditions.

- For Section 3, our main result is that for any fixed $r \geq 3$, the problem r -SWAP is W[1]-hard parameterized by the number of swaps k . Moreover, the same is true for r -SWAP-PROMISE. This should be contrasted with the positive FPT result of Junosza-Szaniawski *et al.* [20] for r -FIX. In

addition, we observe both problems r -FIX and r -SWAP become W[1]-hard for treewidth when the number of colors r is part of the input. The constructions exhibit gadget ideas we use for the sections to follow.

- In Section 4, we prove that under plausible complexity assumptions, r -FIX has no polynomial kernel parameterized by the number of recolorings k , for every $r \geq 3$. We stress that while mentioned as an open problem in [20], the question was subsequently answered by Garnero, Junosza-Szaniawski, Liedloff, Montealegre, and Rzażewski in a full version [13] of [20]. Our result was obtained independently of their work, and uses slightly different ideas.
- Finally, in Section 5, we consider the complexity of the promise variants of the problems (see Section 2 for precise definitions). We show that for $r = 3$, both r -SWAP-PROMISE and r -FIX-PROMISE are NP-hard for planar graphs. Moreover, the problems cannot be solved in $2^{o(\sqrt{n})}$ time unless the Exponential Time Hypothesis fails. On the positive side, using known results, we derive an algorithm for the problem working in $2^{O(\sqrt{n})}$ time.

2 Preliminaries

All graphs in this paper are simple and undirected. For graph-theoretic notion not defined here, we refer the reader to [8]. We write $[n]$ to denote the set $\{1, 2, \dots, n\}$.

2.1 Promises and problem statements

A *promise problem* is a generalization of a decision problem, where the input is guaranteed to belong to a restricted subset among all possible inputs [15].

Definition 1 (Promise problem). *A promise problem is a pair of disjoint sets of strings (S_Y, S_N) , and their union $S_Y \cup S_N$ is called the promise set. An algorithm A decides a promise problem if for every $x \in S_Y$, $A(x) = 1$ and for every $x \in S_N$, $A(x) = 0$; for strings that do not belong to the promise set $x \notin S_Y \cup S_N$ the algorithm A must halt, but can answer arbitrarily.*

A promise problem is in **PromiseNP**, the promise extension of **NP**, if there exists a polynomial p and a polynomial-time verifier V such that for every $x \in S_Y$ there exists y of length at most $p(|x|)$ such that $V(x, y) = 1$ and for every $x \in S_N$ and every y it holds that $V(x, y) = 0$. For a more comprehensive treatment, we refer the reader to [15].

We are then ready to formally define the problems studied in this work. For the promise variants, a coloring c' is said to be a *permutation* of a proper vertex-coloring c if c' can be obtained from c by a finite number of swaps. In other words, the sizes of the color classes of c' match those of an optimal proper coloring.

r -FIX

Instance: A graph $G = (V, E)$, an r -coloring $c : V \rightarrow [r]$, and a positive integer k .

Question: Can c be made into a proper r -coloring of G using at most k recolorings?

r -FIX-PROMISE

Instance: A graph $G = (V, E)$, an r -coloring $c : V \rightarrow [r]$, and a positive integer k .

Promise: $\chi(G) = r$, and c is a permutation of an optimal proper vertex-coloring of G .

Question: Can c be made into a proper r -coloring of G using at most k recolorings?

Note that the number of recolorings needed is precisely the minimum Hamming distance between the given coloring c and a valid coloring c' (if existing).

Similarly, we also define r -SWAP and r -SWAP-PROMISE, where instead of at most k recolorings we have a budget of at most k swaps.

r-SWAP

Instance: A graph $G = (V, E)$, an r -coloring $c : V \rightarrow [r]$, and a positive integer k .

Question: Can c be made into a proper r -coloring of G using at most k swaps?

r-SWAP-PROMISE

Instance: A graph $G = (V, E)$, an r -coloring $c : V \rightarrow [r]$, and a positive integer k .

Promise: $\chi(G) = r$, and c is a permutation of an optimal proper vertex-coloring of G .

Question: Can c be made into a proper r -coloring of G using at most k swaps?

At first glance, the promise conditions might seem to make the two problems *r*-FIX-PROMISE and *r*-SWAP-PROMISE similar: one could think that two recolorings correspond to a swap because if we recolor a vertex, then by the promise, the color must be reinserted elsewhere. However, it is easy to build graphs in which this does not hold. For example, consider a graph G constructed from a triangle with the vertices v_1, v_2, v_3 colored c_1, c_2, c_3 , respectively. Also, connect vertices to G such that v_1 has three pendants (neighbours) colored c_1 and three pendants colored c_2 ; v_2 has three pendants colored c_2 and three pendants colored c_3 ; and v_3 has three pendants colored c_3 and three pendants colored c_1 . Clearly, three recolorings are enough to get a proper coloring. Indeed, we color v_1 with c_3 , v_2 with c_1 , and v_3 with c_2 , and are done. In contrast, in the swap variant, two swaps needed (e.g., swap colors on v_1 and v_3 , and then colors on v_3 and v_2).

2.2 FPT-reductions and (kernelization) lower bounds

In this subsection, we briefly review the necessary basics of parameterized complexity.

Definition 2. Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. A parameterized reduction from A to B is an algorithm such that given an instance (x, k) of A , it outputs an instance (x', k') of B such that

1. (x, k) is a YES-instance of A iff (x', k') is a YES-instance of B ,
2. $k' \leq g(k)$ for some computable function g , and
3. the running time is $f(k) \cdot |x|^{O(1)}$ for some computable function f .

In the CLIQUE problem, we are given a graph G and an integer k . The task is to decide whether G contains a complete subgraph on k vertices. The class of problems reducible to CLIQUE under parameterized reductions is denoted by $W[1]$. We define hardness and completeness analogously to classical complexity, but assume parameterized reductions. That is, a problem is said to be $W[1]$ -hard if CLIQUE (and thus each problem in $W[1]$) can be reduced to it by a parameterized reduction. It is widely believed that $FPT \neq W[1]$.

Let us recall the well-known Exponential Time Hypothesis (ETH), which is often the assumption used for excluding the existence of algorithms that are considerably faster than e.g., brute-force.

Conjecture 3 (Exponential Time Hypothesis [16]). *There exists a constant $c > 0$, such that there is no algorithm solving 3-SAT in time $O^*(2^{cn})$, where n is the number of variables.*

Suppose φ is an instance of 3-SAT with n variables and m clauses. It holds that if there is a linear reduction from 3-SAT to, say, a graph problem X , then the problem X cannot be solved in time $2^{o(n'+m')}$, where n' and m' denote the number of vertices and edges, respectively. Similar reasoning can be applied for $W[1]$ -hard problems. For instance, it is known that there is no $f(k)n^{o(k)}$ -time algorithm for INDEPENDENT SET for any computable function f , unless ETH fails. Then the existence of a parameterized reduction with a linear parameter dependence from INDEPENDENT SET to a problem X' implies a lower bound for X' under ETH. For more examples and discussion, we refer the reader to [6, 25].

Finally, let us then mention the machinery we use to obtain kernelization lower bounds later on (for more details, see [2, 6]).

Definition 4. An equivalence relation \mathcal{R} on Σ^* is a polynomial equivalence relation if (1) there is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $\mathcal{R}(x, y)$ in $(|x| + |y|)^{O(1)}$ time; and (2) for any finite set $S \subseteq \Sigma^*$ the equivalence relation \mathcal{R} partitions the elements of S into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.

Definition 5 (Cross-composition). Let $L \subseteq \Sigma^*$ and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say L cross-composes into Q if there is a polynomial equivalence relation \mathcal{R} and an algorithm which, given t strings x_1, x_2, \dots, x_t belonging to the same equivalence class of \mathcal{R} , computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^t |x_i|$ such that (1) $(x^*, k^*) \in Q$ iff $x_i \in L$ for some $i \in [t]$; and (2) k^* is bounded polynomially in $\max_{i=1}^t |x_i| + \log t$.

Theorem 6 ([2]). Assume that an NP-hard language L cross-composes into a parameterized language Q . Then Q does not admit a polynomial kernel, unless $\text{NP} \subseteq \text{coNP/poly}$ and the polynomial hierarchy collapses.

3 Parameterized aspects of restoring corrupted colorings

Junosza-Szaniawski *et al.* [20] focused on the r -FIX problem, that is, the number of colors in the coloring is fixed to be r . Among other results, they showed the problem is FPT parameterized by treewidth. In other words, the FIX problem (same as r -FIX but r is part of the input) is FPT for the combined parameter $(r + t)$, where t is the treewidth of the input graph. In contrast, we observe the problem is W[1]-hard when the parameter is only treewidth.

In the PRECOLORING EXTENSION problem (PREXT), we are given a graph $G = (V, E)$, a set $W \subseteq V$ of precolored vertices, and a precoloring $c : W \rightarrow [r]$ of the vertices in W . The goal is to decide whether there is a proper r -coloring c' of G extending the coloring c (i.e., $c'(v) = c(v)$ for every $v \in W$). When r is fixed, we call the problem r -PRECOLORING EXTENSION (r -PREXT). Let us then proceed with the following observation.

Lemma 7. There exists a polynomial time algorithm which given an instance $I = (G, W, c, r)$ of PRECOLORING EXTENSION constructs an instance $I' = (G', r', c', k')$ of FIX, such that I is a YES-instance of PRECOLORING EXTENSION iff I' is a YES-instance of FIX.

Proof. Let $I = (G, W, c, r)$ be an instance of PRECOLORING EXTENSION. To obtain, in polynomial time, an instance $I' = (G', r', c', k')$ of FIX, we proceed as follows. First, let $G' = G$, $r' = r$, and set the number of recolorings $k' = |V \setminus W|$. Then to each precolored vertex $w \in W$, we attach $(r' - 1) \cdot (k' + 1)$ pendant vertices, called P_w . Build c' from c as follows. We color vertices in P_w such that there are precisely $k' + 1$ vertices colored in every color $c \in ([r'] \setminus \{c(w)\})$. We retain the colors on the vertices in W , and color each uncolored vertex with color 1. Observe that k' recolorings will not suffice to change the color of $w \in W$ as it has $k' + 1$ pendants colored in each color distinct from $c(w)$. Thus, it is easy to see $I = (G, W, c, r)$ is a YES-instance of PRECOLORING EXTENSION iff $I' = (G', r', c', k')$ is a YES-instance of FIX. \square

To make the result hold for SWAP, we add $r \cdot k'$ isolated vertices and color them so that we provide a choice of one of the r colors for each of the k' non-precolored vertices. Finally, it is well-known the addition of vertices of degree at most 1 does not increase the treewidth of a graph. As PRECOLORING EXTENSION is W[1]-hard for treewidth [10], we obtain the following.

Corollary 8. Both problems FIX and SWAP are W[1]-hard parameterized by treewidth.

Because PRECOLORING EXTENSION is NP-complete when restricted to distance-hereditary graphs [3] (and thus for e.g., chordal graphs), we immediately observe the following.

Corollary 9. Both problems FIX and SWAP are NP-complete when restricted to the class of distance-hereditary graphs.

The above also implies hardness for bounded cliquewidth graphs.

Junosza-Szaniawski *et al.* [20] proved that for every fixed r , the problem r -FIX is FPT parameterized by the number of recolorings. However, when the basic operation is a swap instead of a recoloring,

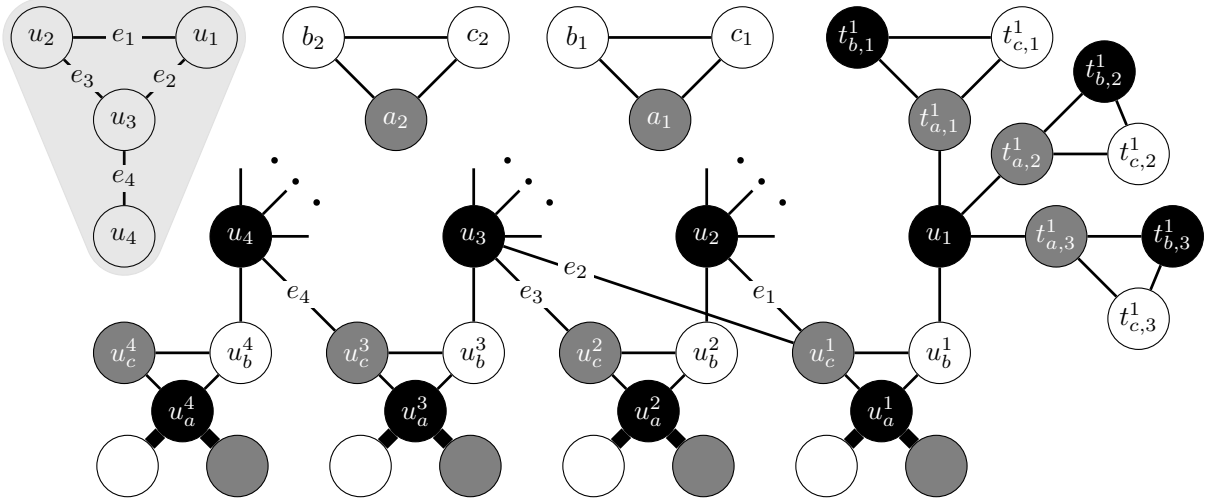


Figure 1: An instance (G, k) of INDEPENDENT SET with $k = 2$ (highlighted in gray) transformed to an instance of r -SWAP. Thick edges at the bottom correspond to $k + 1$ pendant vertices. A gadget has been expanded for u_1 : the three dots hide a similar gadget for each of u_2 , u_3 , and u_4 . The colors are “● = 1”, “○ = 2”, and “● = 3”.

the problem becomes hard. This is established by the following lemma. For the result, we give a parameterized reduction from the well-known INDEPENDENT SET problem. In this problem we are given a graph $G = (V, E)$, and an integer k . The goal is to decide whether G contains a set of k pairwise non-adjacent vertices. The problem is well-known to be $W[1]$ -hard parameterized by k .

Lemma 10. *There exists a polynomial time algorithm which given an instance $I = (G, k)$ of INDEPENDENT SET constructs an instance $I' = (G', r, c, k')$ of r -SWAP for $r = 3$ such that I is a YES-instance of INDEPENDENT SET iff I' is a YES-instance of r -SWAP.*

Proof. Let $V(G) = \{u_1, u_2, \dots, u_n\}$. To construct the graph G' , begin with $V(G') = V(G)$ and $k' = 2k$. Add k disjoint triangles $\{a_1, b_1, c_1\}, \dots, \{a_k, b_k, c_k\}$, and color them such that $c(a_j) = 3$ and $c(b_j) = c(c_j) = 2$, for $j \in [k]$. For each $i \in [n]$, add a disjoint triangle $C_i = \{u_a^i, u_b^i, u_c^i\}$. In each, color $c(u_a^i) = 1$, $c(u_b^i) = 2$, and $c(u_c^i) = 3$. Attach $2(k + 1)$ pendant vertices to u_a^i and color $k + 1$ of them with color 2, and $k + 1$ of them with color 3. For each i , add the edge $u_i u_b^i$. For every $i, j \in [n]$, if $j > i$ and $u_i u_j \in E(G)$, add the edge $u_j u_c^i$. Finally, for each $i \in [n]$, add $k + 1$ disjoint triangles $\{t_{a,1}^i, t_{b,1}^i, t_{c,1}^i\}, \dots, \{t_{a,k+1}^i, t_{b,k+1}^i, t_{c,k+1}^i\}$. These $k + 1$ disjoint triangles are colored such that $c(t_{a,j}^i) = 3$, $c(t_{b,j}^i) = 2$, and $c(t_{c,j}^i) = 1$, where $j \in [k + 1]$. Add the edge $u_i t_{a,j}^i$, for $i \in [n]$ and $j \in [k + 1]$. This completes the construction of G' . An example is shown in Figure 1. Let us then prove $I = (G, k)$ is a YES-instance of INDEPENDENT SET iff $I' = (G', r, c, k')$ is a YES-instance of r -SWAP.

Let $S = \{s_1, \dots, s_k\} \subseteq V(G)$ be an independent set. By construction, there are exactly k conflicts in G' between b_j and c_j , for $j \in [k]$. Swap c_j with u_j , spending a total of k swaps. By doing so, we fixed k conflicts but also introduced k new conflicts. In particular, the new conflicts are between u_i and u_b^i , as they are both colored 2. We swap u_b^i with u_c^i (colored 3), and claim G' is properly colored. By construction, u_c^i is adjacent to $u \in V(G)$ only if u and u_i are adjacent in $V(G)$. As S is an independent set, each $u \in V(G)$ adjacent to u_c^i is colored 1. Thus, c is a proper coloring for G' , and we are done.

For the other direction, suppose $k' = 2k$ swaps suffice to obtain a proper vertex coloring from c . Again, each of the k conflicts between b_j and c_j , for $j \in [k]$, must be fixed. To resolve the conflicts, we must swap either b_j or c_j with a vertex colored 1. Without loss, suppose we choose c_j over b_j . Observe that for every $i \in [n]$, we cannot swap u_a^i with any vertex as it has $k + 1$ pendant vertices colored 2, and $k + 1$ pendant vertices colored 3. Thus, there are two possibilities: either we swap c_j with u_i , or with $t_{a,j}^i$, for some $i \in [n]$ and $j \in [k + 1]$. If the swap occurs between c_j and $t_{a,j}^i$, we must then swap $t_{b,j}^i$ with u_i for some $i \in [n]$. This introduces a conflict between the chosen u_i and its adjacent vertex u_b^i . After we fix this conflict, we have used a total of 3 swaps, totaling $3k > k'$ swaps if we followed this strategy for each c_j . Thus, as $k' = 2k$ swaps suffice, and we are looking for an independent set of size exactly k , we must swap each of the vertices c_j with some u_i . Clearly, two vertices u_i, u_ℓ , for $i, \ell \in [n]$, adjacent in

G cannot be used to fix the conflicts, for otherwise we would have a conflict between the vertices u_b^i and u_b^{ℓ} (both colored 2). We conclude that the vertices u_i a vertex c_j is swapped with form an independent set. \square

By adding a properly colored r -clique to the constructed graph, we can extend the lemma to cover every fixed value of r . Thus, we have the following.

Theorem 11. *For every $r \geq 3$, the problem r -SWAP is $W[1]$ -hard parameterized by the number of swaps k . Furthermore, there is no $f(k)n^{o(k)}$ -time algorithm for the problem unless ETH fails, where f is a computable function.*

In addition, it is straightforward to extend the construction of Lemma 10 to show the promise variant, namely r -SWAP-PROMISE, is $W[1]$ -hard parameterized by the number of swaps k .

Corollary 12. *For every $r \geq 3$, the problem r -SWAP-PROMISE is $W[1]$ -hard parameterized by the number of swaps k . Furthermore, there is no $f(k)n^{o(k)}$ -time algorithm for the problem unless ETH fails, where f is a computable function.*

Proof. Assume the construction of an instance $I = (G', r, c, k')$ of r -SWAP of Lemma 10. To prove the claim, it suffices to augment the construction to ensure the promise holds, i.e., that with a finite number of swaps we have that G' is properly r -colored and $\chi(G') = r$.

A double star S'_n is the complete bipartite graph $K_{1,n}$ with each edge subdivided. Formally, $S'_n = (\{s\} \cup \{q_1, \dots, q_n\} \cup \{q'_1, \dots, q'_n\}, \{(s, q_i), (q_i, q'_i) \mid i \in [n]\})$. To enforce the promise, add n disjoint double stars S'_{k+1} to G' . Each double star is colored such that the central vertex s receives color 1, vertices q_i adjacent to the central vertex color 2, and the remaining vertices q'_i color 3. Observe that after swapping the central vertex s (colored 1) of a S'_{k+1} with (say) b_1 (colored 2), we require $k' + 1$ more swaps to fix the conflicts residing at that particular S'_{k+1} . Nevertheless, given enough swaps, $k' \cdot (k' + 1)$ to be precise, we can properly r -color G' . Finally, to guarantee $\chi(G') = r$, it suffices to add a disjoint properly r -colored clique. \square

4 No polynomial kernel for r -FIX

Junosza-Szaniawski *et al.* [20] showed that for any fixed r , the problem r -FIX is FPT parameterized by the number of recolorings k . In particular, their result implies a kernel of exponential size for the problem. Thus, they asked whether or not there is a kernel of polynomial size. The question was answered in the negative in a full version of [20] by Garnero *et al.* [13]. Independently of their work, in what is to follow, we give an alternative proof of the theorem.

Lemma 13. *For $r = 3$, the problem r -FIX parameterized by the number of recolorings k does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP/poly}$.*

Proof. We show that 3-SAT cross-composes into r -FIX parameterized by the number of recolorings k . By choosing an appropriate polynomial equivalence relation \mathcal{R} , we can assume we are given a sequence $\varphi_1, \varphi_2, \dots, \varphi_t$ of 3-SAT instances with an equal number of variables, denoted by n , and an equal number of clauses, denoted by m .

Let us then proceed with a cross-composition algorithm that composes t input instances $\varphi_1, \varphi_2, \dots, \varphi_t$ which are equivalent under \mathcal{R} into a single instance of r -FIX parameterized by the number of recolorings. Specifically, we construct an instance (G, k) of r -FIX, where G is a vertex-colored graph, and k the number of recolorings. Our plan is to convert each 3-SAT instance $\varphi_1, \varphi_2, \dots, \varphi_t$ to an instance $\varphi'_1, \varphi'_2, \dots, \varphi'_t$ of 4-SAT. For each resulting instance of 4-SAT, we apply the standard reduction from 4-SAT to 3-COLORING (see e.g., [12]). Finally, the resulting graphs are connected by a *spread gadget*, which acts as an instance selector. Let us first describe the gadgets, and then the construction of the whole graph G . At the same time, we describe a 3-coloring $c : V(G) \rightarrow [3]$. We set $k = 2 \log_2(t) + 2n + 9m$. Our construction depends crucially on k , and its choice will become apparent later on.

3-SAT to 4-SAT. For each 3-SAT formula φ_h , where $h \in [t]$, introduce a new variable u_h , and add it to each clause of φ_h . We call the resulting 4-SAT formula φ'_h . Observe that by setting u_h to true we satisfy φ'_h .

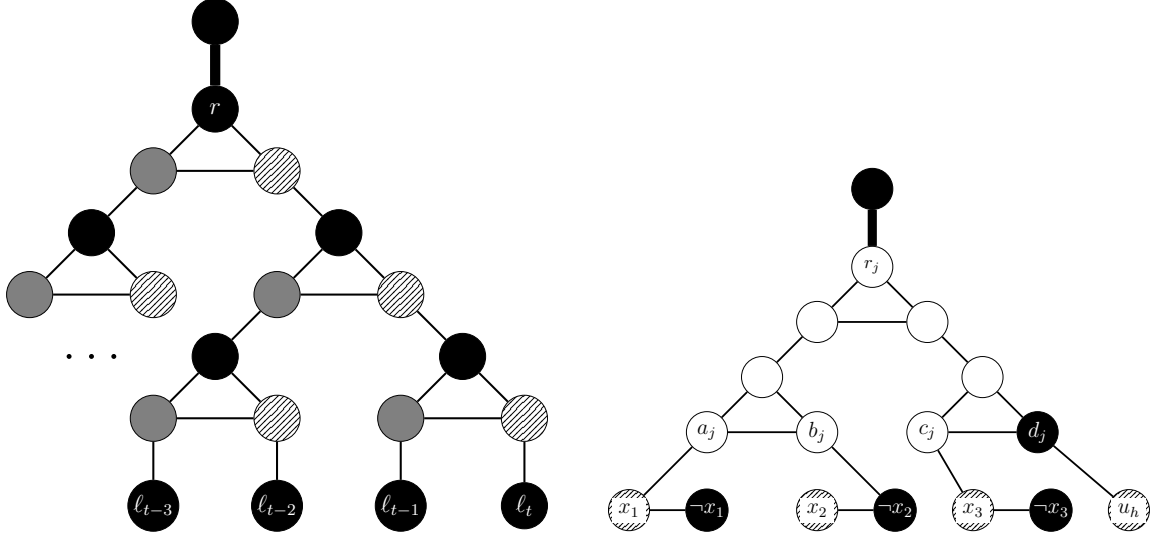


Figure 2: **(a)** In the spread gadget, fixing the conflict at the root r corresponds to choosing an instance ℓ_h , for $h \in [t]$. **(b)** The clause gadget with its initial vertex-coloring, where the white vertices are colored such that (P_1) and (P_2) are respected. In both figures, thick edges correspond to $k + 1$ pendant vertices. The colors are “ $\bullet = 1$ ”, “ $\textcircled{\diagup} = 2$ ”, and “ $\textcircled{\diagdown} = 3$ ”.

The variable vertices. Let the n variables of φ_h , where $h \in [t]$, be $x_{1,h}, x_{2,h}, \dots, x_{n,h}$. We introduce n disjoint 2-cliques labelled $\{x_{1,h}, \neg x_{1,h}\}, \dots, \{x_{n,h}, \neg x_{n,h}\}$. Set $c(x_{i,h}) = 2$ and $c(\neg x_{i,h}) = 1$, for $i \in [n]$ (i.e., initially we set each variable true). Add an isolated vertex u_h , and let $c(u_h) = 1$. We refer to each of the $2n + 1$ vertices as *variable vertices*.

The clause gadget. Denote by $C_{h,j}$ the j th clause of φ'_h . For each clause $C_{h,j}$, where $h \in [t]$ and $j \in [m]$, construct the following clause gadget $H_{h,j}$ (see Figure 2 (b)). Take three disjoint triangles $\{a_j, b_j, y_{1,j}\}, \{c_j, d_j, y_{2,j}\}, \{y_{3,j}, y_{4,j}, r_j\}$, and add the edges $y_{1,j}y_{4,j}, y_{2,j}y_{3,j}$, and $y_{5,j}r_j$. We add $k + 1$ pendant vertices adjacent to r_j and color them with color 1. This guarantees k recolorings cannot give r_j color 1. Vertices a_j, b_j, c_j and d_j correspond to the 4 literals each clause has. Thus, we connect them to the corresponding variable vertices. That is, when $u \in \{a_j, b_j, c_j, d_j\}$ corresponds to the variable $x_{i,h}$, $i \in [n]$, we add the edge $ux_{i,h}$ (and similarly when its negated). The following properties hold for a clause gadget $H_{h,j}$.

- (P_1) If all four variable vertices of $H_{h,j}$ have color 1, then r_j must have color 1 (costing $k + 1$ recolorings to properly color the gadget).
- (P_2) The gadget $H_{h,j}$ can be properly 3-colored if one of the attached variable vertices (including u_h) have color 2.

The spread gadget. The spread gadget is constructed by starting from a complete binary tree on t leaves $\ell_1, \ell_2, \dots, \ell_t$ with the root r . We replace each internal vertex with a triangle, and attach $k + 1$ pendant vertices to r . Thus, the distance from r to any leaf is $2 \log_2(t)$. We color root r , its pendant vertices, and each leaf $\ell_1, \ell_2, \dots, \ell_t$ with color 1. In a triangle, the top vertex receives color 1, the right vertex color 2, and the left vertex color 3 (see Figure 2 (b)). This finishes the construction of the spread gadget.

To obtain G , we connect the described gadgets together as follows. For each φ'_h , where $h \in [t]$, we add a vertex w_h . Make w_h adjacent to each variable vertex, and set $c(w_h) = 3$. We give w_h altogether $2(k + 1)$ pendant vertices, and color them so that $k + 1$ of them have color 1, and $k + 1$ of them have color 2. This enforces $c(w_h) = 3$ if only k recolorings are available. In the spread gadget, each leaf ℓ_h is made adjacent to both u_h and w_h .

This completes the construction of the graph G . Recall $k = 2 \log_2(t) + 2n + 9m$, and output the instance (G, k) of r -FIX. Let us then prove that (G, k) is a YES-instance of r -FIX iff one of φ_h is satisfiable for $h \in [t]$.

Correctness. Suppose (G, k) is a YES-instance of r -FIX. By construction, the root r and its $k + 1$ pendant vertices are colored with color 1 in the spread gadget. As we only have a budget of k recolorings, we must recolor r . By doing so, we introduce a conflict into the triangle containing r . When this conflict is fixed, we move it to one of the two succeeding triangles. Further continuing to fix the conflict, we propagate it down to one of the leaves ℓ_s , for some $s \in [t]$. Intuitively, the propagation to ℓ_s means we have chosen to solve the instance φ_s . By construction, ℓ_s forms a triangle with u_h (colored 2) and w_h (colored 3). As w_h has $k + 1$ pendants colored with 1 and $k + 1$ pendants colored with 2, we must move the conflict to u_h . By moving the conflict from r to u_h , we used precisely $2 \log_2(t)$ swaps. Now, u_h has color 1, as do all the vertices in $D = \{d_j \in V(C_{s,j}) \mid j \in [m]\}$. As $c(u_h) = 1$, we have set the truth value of u_h to false. Thus, the truth value of φ_s is not affected by the truth value of u_h . As $|D| = m$, each vertex in D can be recolored in m recolorings. Moreover, $9m$ recolorings suffice to swap the color of each vertex in the three vertex-disjoint triangles a clause gadget has. By construction, the initial vertex-coloring corresponds to a truth assignment $\tau = \{z_1, z_2, \dots, z_n\} \in \{1\}^n$ setting each variable to true. Clearly, $2n$ recolorings suffice to reverse τ , i.e., change τ to τ' such that the Hamming distance of τ and τ' is n . Therefore, by (P_2) , if (G, k) is a YES-instance of r -FIX, then φ_s is satisfiable.

For the other direction, suppose φ_s is satisfiable for some $s \in [t]$. Again, the initial vertex-coloring corresponds to a truth assignment $\tau = \{z_1, z_2, \dots, z_n\} \in \{1\}^n$ setting each variable to true. Using at most $2n + 9m$ recolorings, we turn the initial vertex-coloring to a vertex-coloring corresponding to $\tau' = \{z'_1, z'_2, \dots, z'_n\} \in \{0, 1\}^n$ such that φ_s is satisfied under τ' . Indeed, observe that as φ_s is satisfiable, (P_1) is not violated. Moreover, observe that τ' satisfies φ_s regardless of the truth value of u_s . Thus, we can freely let $c(u_s) = 1$. But now we introduce a conflict between ℓ_s (colored 1) and its unique neighbor in the spread gadget. However, using precisely $2 \log_2(t)$ recolorings, we propagate this conflict, and in particular the color 1, up to the root r . At most $k = 2 \log_2(t) + 2n + 9m$ recolorings have been used, and no conflicts remain in G . Thus, if at least one of $\varphi_1, \dots, \varphi_t$ is a YES-instance of 3-SAT, then (G, k) is a YES-instance of r -FIX. This concludes the proof. \square

In order to extend the above result to hold for every $r \geq 4$, we attach $(r - 3) \cdot (k + 1)$ pendant vertices to each vertex of the construction. These pendant vertices are colored in the obvious way such that each “original” vertex must receive exactly one of the colors 1, 2, or 3.

Theorem 14 ([13]). *For every $r \geq 3$, the problem r -FIX parameterized by the number of recolorings k does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP/poly}$.*

Note that in the light of Theorem 11, the existence of a kernel of any size depending only on k and r for r -SWAP is highly unlikely.

5 Chromatic villainy: r -SWAP-PROMISE is hard

As the main result of this section, we prove that 3-SWAP-PROMISE is NP-hard when restricted to the class of planar graphs. In other words, even with the additional information that some proper vertex-coloring is always obtainable after a finite number of swaps (and no other proper vertex-coloring with less than 3 colors exists), the problem remains hard.

Our reduction will be from the r -PREXT problem, shown to be NP-complete for $r = 3$ when restricted to bipartite planar graphs by Kratochvíl [22]. In fact, although not explicitly stated, the following slightly stronger result is obtained from [22].

Theorem 15 ([22]). *The r -PREXT problem is NP-complete for $r = 3$ when restricted to the class of bipartite planar graphs, and each precolored vertex has degree 1, that is, $\deg(w) = 1$ for every $w \in W$.*

The reader should be aware that in the following, we use the color set $\{0, 1, 2\}$ instead of $[3]$. This will make it more convenient to describe the coloring through modular arithmetic. We are then ready to proceed with the main result of the section.

Theorem 16. *3-SWAP-PROMISE is NP-hard when restricted to the class of planar graphs. Moreover, the same is true even when every swap must be between adjacent vertices.*

Proof. Let $(G = (V, E), W, c)$ be an instance of r -PREXT, where G is an n -vertex bipartite planar graph, $W \subseteq V$ a set of precolored vertices, and $c : W \rightarrow \{0, 1, 2\}$ a precoloring of the vertices in W .

By Theorem 15, we may assume without loss that each (precolored) vertex in W has degree 1. Our construction crucially depends on this fact. Let $r = 3$ be a fixed color bound, and let $h = |V \setminus W|$. We will construct a graph H along with its vertex-coloring c_H such that the precoloring c can be extended to a valid r -coloring of G iff at most h swaps are needed to transform c_H to an optimal proper vertex-coloring of H , i.e., $B(c_H) \leq h$. To enforce the promise of the problem, it shall hold for c_H that (i) it uses precisely $\chi(H)$ colors, and that (ii) by using a finite number of swaps c_H can be transformed into a proper coloring of H .

Construction. Let $X = V \setminus W$ be the set of uncolored vertices, let A and B be the bipartition of G , that is, $V = A \cup B$, and let us name the set of r colors $C = \{0, 1, 2\}$. The graph H and its vertex-coloring c_H are constructed from G and its precoloring c as follows.

- We retain the coloring on the vertices of W , that is, $c_H(w) = c(w)$, for every $w \in W$.
- If $x \in X$ has one or more neighbors colored with color i (observe it cannot have distinctly colored neighbors), we set $c_H(x) = i$. If all neighbors of x are uncolored, we set $c_H(x) = 0$ if $x \in A$. Otherwise, $x \in B$, so we set $c_H(x) = 1$. For each x , we also add two vertices x_1 and x_2 along with the edges xx_1 and xx_2 . We color $c_H(x_1) = (i + 1) \bmod 3$ and $c_H(x_2) = (i + 2) \bmod 3$.
- For each precolored vertex $w \in W$ we add $2(h+1)$ new vertices $s_{w,1}, \dots, s_{w,h+1}$ and $t_{w,1}, \dots, t_{w,h+1}$. These will be made pendant vertices of w by adding the altogether $2(h+1)$ edges $ws_{w,\ell}$ and $wt_{w,\ell}$ where $\ell \in [h+1]$. They receive a color as follows, where $c(w)$ denotes the color of w :
 - if $w \in A \wedge c(w) \neq 0$, then $c_H(s_{w,\ell}) = 0$ and $c_H(t_{w,\ell}) = f$, where $f \in C \setminus \{0, c(w)\}$;
 - if $w \in B \wedge c(w) \neq 1$, then $c_H(s_{w,\ell}) = 1$ and $c_H(t_{w,\ell}) = g$, where $g \in C \setminus \{1, c(w)\}$; and
 - in all other cases $c_H(s_{w,\ell}) = (i + 1) \bmod 3$, and $c_H(t_{w,\ell}) = (i + 2) \bmod 3$.
- For every precolored vertex $w \in A \wedge c(w) \neq 0$, we add h new vertices $s'_{w,1}, \dots, s'_{w,h}$ with the edges $ws_{w,j}s'_{w,j}$, and set $c_H(s'_{w,j}) = c(w)$, where $j \in [h]$.
- For every precolored vertex $w \in B \wedge c(w) \neq 1$, we add h new vertices $s'_{w,1}, \dots, s'_{w,h}$ with the edges $ws_{w,j}s'_{w,j}$, and set $c_H(s'_{w,j}) = c(w)$, where $j \in [h]$.
- Finally, consider an arbitrary precolored vertex $w \in W$, and its set of $h+1$ pendant vertices $t_{w,j}$. We choose an arbitrary vertex among the $t_{w,j}$ vertices, and call it v . Then, we add two vertices r and r' along with the edges vr , vr' , and rr' . These vertices are colored such that $c_H(r) = (c_H(v) + 1) \bmod 3$ and $c_H(r') = (c_H(v) + 2) \bmod 3$.

This finishes the construction of the graph H along with its vertex-coloring c_H . It is straightforward to verify G is planar, but not bipartite because of the triangle on the vertices v , r , and r' . An example is shown in Figure 3. We will then prove $(G = (V, E), W, c)$ is YES-instance of PREXT iff $B(c_H) \leq h$, that is, if h swaps suffice to transform c_H into a proper coloring of H .

Correctness. Suppose c can be extended to a proper vertex-coloring c' of G . We will show h swaps suffice to transform c_H into a proper coloring of H . For each $x \in X$, we perform a swap between x and either x_1 or x_2 . Then for every x it holds that either $c_H(x) = c'(x)$ (and there is no need to swap it), or one of the two described swaps can change the color of x to $c'(x)$. Now, let $c_H(x)$ be the color of x before the swap, and $c'_H(x) = c'(x)$ its color after the swap. Let u_1, \dots, u_m be the neighbors of x . If $u_i \in W$ was a precolored vertex, then $c_H(u_i) = c_H(x) \neq c'_H(x)$ by construction; if $u_i \in X$ was an uncolored vertex then the valid coloring c' guarantees that $c'_H(x) \neq c'_H(u_i)$. Thus, the claim follows.

For the other direction, suppose $B(c_H) \leq h$. Consider a precolored vertex $w \in W$ and let $c_H(w) = i$. We claim that for any valid extension c' of c , it holds that $c'(w) = c_H(w) = c(w)$. More precisely, we will show that if the color of w was changed, then it is impossible for c' to be an extension of c . By construction, the vertex w has $h+1$ neighbors $s_{w,\ell}$ each colored p , and $h+1$ neighbors $t_{w,\ell}$ each colored q with $i \neq p \neq q$. Thus, if one swap was used to change the color on w , then after $h-1$ swaps there would be at least one edge incident to w with its endpoints having the same color. So we have that $c'(w) = c_H(w) = c(w)$. Moreover, c' is completed to an extension of c by picking the colors $c'_H(x)$ assigned to the uncolored vertices $x \in X$ after the h swaps. This completes the proof of correctness for our reduction.

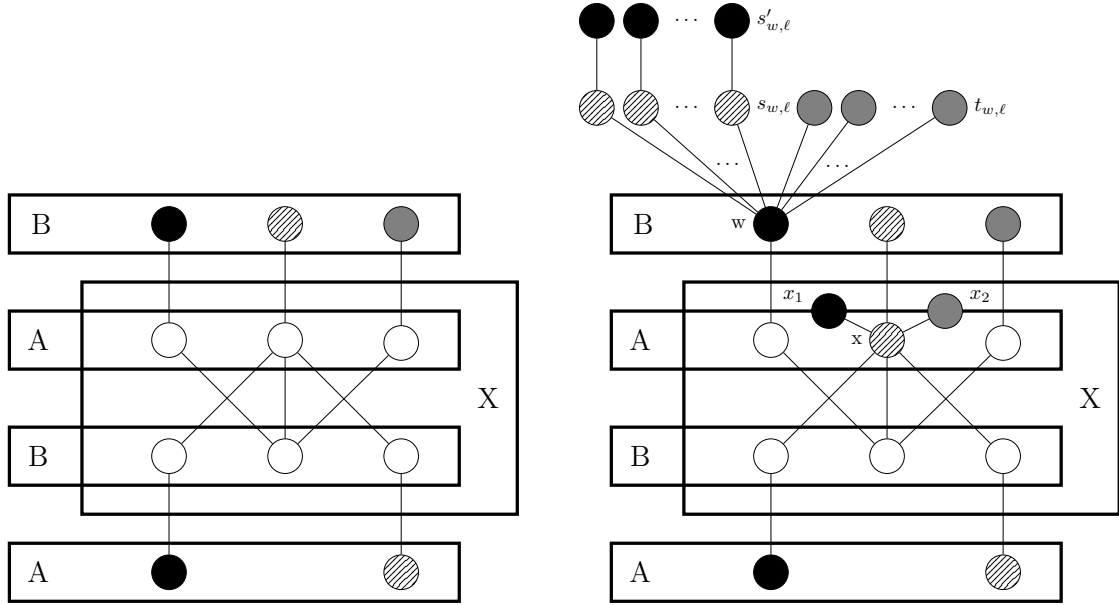


Figure 3: (a) A partially precolored input graph G of r -PR-EXT. (b) To reduce clutter, the reduction of Theorem 16 expanded for only two vertices $x \in X$ and $w \in W$. The colors are “● = 0”, “▨ = 1”, and “● = 2”.

Promise. Let us then show that the promise holds as well. That is, we show that c_H can be transformed into a proper vertex-coloring c''_H of H with a finite number of swaps even if the original precoloring of G cannot be extended to a proper coloring (but in this case, more than h swaps are needed).

First, we show that a finite number of swaps gives us a 2-coloring for $A \cup B$ such that every vertex in A receives color 0, and every vertex in B color 1. Afterwards, we will adjust the remaining pendant vertices $s_{w,\ell}$, $t_{w,\ell}$, and $s'_{w,j}$ so that no color conflict remains.

If $x \in X \cap A \wedge c_H(x) \neq 1$, then we swap it with one of its neighbors x_1 or x_2 and get $c''_H(x) = 0$. Similarly, if $x \in X \cap B \wedge c_H(x) \neq 2$, a swap with either x_1 or x_2 gives us $c''_H(x) = 1$.

Let us then consider the precolored vertices. If $w \in W \cap A \wedge c_H(w) \neq 0$, we swap w with $s_{w,h+1}$ which is colored 0. This causes a conflict $c''_H(w) = c_H(s_{w,j}) = 0$, which is fixed by swapping $s_{w,j}$ with $s'_{w,j}$ that are colored $c_H(s'_{w,j}) = c_H(w) \neq 0$. Similarly, if $w \in W \cap B \wedge c_H(w) \neq 1$, we swap w with $s_{w,h+1}$ which is colored 1. This causes conflicts $c''_H(w) = c_H(s_{w,j}) = 1$, where $j \in [h]$. To fix them, we swap $s_{w,j}$ with $s'_{w,j}$ that are colored $c_H(s'_{w,j}) = c_H(w) \neq 1$. Thus, we have that all vertices in A are colored 1, and all vertices in B are colored 1. Moreover, for all $w \in W$ we have $c''_H(w) \neq c''_H(s_{w,\ell}) \wedge c''_H(w) \neq c''_H(t_{w,\ell})$, where $\ell \in [h+1]$. Also, $c''_H(s_{w,j}) \neq c''_H(s_{w,j})$, where $j \in [h]$. Thus, c''_H is a valid coloring of H and the triangle v, r, r' guarantees that $\chi(H) = 3$. Thus, the claim follows. \square

By removing the promise condition, we can modify our reduction to obtain the following.

Corollary 17. *For every $r \geq 3$, the problem r -SWAP is NP-complete for bipartite planar graphs.*

Another corollary follows by a chain of reductions. First, Lichtenstein [24] gives a reduction from 3-SAT to PLANAR 3-SAT showing PLANAR 3-SAT cannot be solved in time $2^{o(\sqrt{n+m})}$, unless ETH fails. Continuing to compose reductions, Mansfield [26] gives a linear reduction from PLANAR 3-SAT to PLANAR 1-IN-3-SAT, which is then similarly reduced by Kratochvíl [22] to r -PR-EXT (the result of Theorem 15). Finally, it can be verified the construction of Theorem 16 has linear size, giving us the following.

Corollary 18. *There is no algorithm which solves PLANAR 3-SWAP-PROMISE in $2^{o(\sqrt{n})}$ time unless ETH fails.*

However, on a positive side, we claim that for any fixed r , the problem PLANAR r -FIX (and its promise variant) can be solved in $2^{O(\sqrt{n})}$ time. To see this, we recall that Junosza-Szaniawski *et al.* [20] showed that for any fixed r , the optimization variant of r -FIX is solvable in $O(nr^{t+2})$ time on graphs of

treewidth t . To leverage this result, it is enough to recall the treewidth of a planar graph is $O(\sqrt{n})$. This implies a $2^{O(\sqrt{n})}$ -time algorithm for PLANAR r -FIX.

Finally, let us mention that by modifying the construction of Theorem 16 slightly, one can show a similar result for r -FIX-PROMISE, and its non-promise variant as well.

Theorem 19. *3-FIX-PROMISE is NP-hard when restricted to the class of planar graphs.*

Proof. Consider the construction in Theorem 16. For each $x \in X$, remove the edges xx_1 and xx_2 , and add the edge x_1x_2 .

Observe that it still holds that in h recolorings, it is impossible to recolor a precolored vertex $w \in W$. Thus, the correctness of the reduction holds. To see that the promise is enforced, note that it is still true that $\chi'(G) = 3 = r$. For each $x \in X$, there is a corresponding 2-clique, from which a color distinct from $c'(x)$ can be swapped for x . Thus, the promise is also enforced. We have a valid reduction, and thus conclude the proof. \square

By relaxing the requirement on the promise condition, we establish again the following. We also note that using different ideas, the same conclusion was reached in [13].

Corollary 20 ([13]). *For every $r \geq 3$, the problem r -FIX is NP-complete for bipartite planar graphs.*

As also remarked in [13], it is interesting to contrast the above with the results of [20] where it was shown that when $r = 2$, the problem r -FIX is solvable in polynomial time. In other words, if we have a bipartite graph that is *not* colored optimally (i.e. more than two colors are used), fixing the coloring is hard.

6 Conclusions

We further investigated the complexity of restoring corrupted colorings, especially from a parameterized perspective. Interestingly, we showed that r -SWAP is W[1]-hard parameterized by the number of swaps, while r -FIX is known to be FPT parameterized by the number of recolorings. We believe the problems behave similarly for treewidth. Indeed, we conjecture that r -SWAP is W[1]-hard parameterized by treewidth, for every $r \geq 3$. One could also consider other natural basic operations, such as swaps between adjacent vertices.

Finally, it might be interesting to perform a similar study for edge-colored graphs. In particular, how does the complexity of edge recoloring compare to vertex recoloring?

References

- [1] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.
- [2] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal of Discrete Mathematics*, 28(1):277–305, 2014.
- [3] F. Bonomo, G. Durán, and J. Marenco. Exploring the complexity boundary between coloring and list-coloring. *Annals of Operations Research*, 169(1):3–16, 2008.
- [4] P. S. Bonsma, A. E. Mouawad, N. Nishimura, and V. Raman. The complexity of bounded length graph recoloring and CSP reconfiguration. In *Proceedings of the 9th International Symposium on Parameterized and Exact Computation, IPEC 2014, Wrocław, Poland, September 10-12*, pages 110–121, 2014.
- [5] S. A. Clark, J. E. Holliday, S. H. Holliday, P. Johnson, J. E. Trimm, R. R. Rubalcaba, and M. Walsh. Chromatic villainy in graphs. *Congressus Numerantium*, 182:171, 2006.
- [6] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

- [7] M. de Berg, K. Buchin, B. M. P. Jansen, and G. J. Woeginger. Fine-grained complexity analysis of two classic TSP variants. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, Rome, Italy, July 11-15*, pages 5:1–5:14, 2016.
- [8] R. Diestel. *Graph Theory*. Springer-Verlag Heidelberg, 2010.
- [9] S. Even, A. L. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, 1984.
- [10] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.
- [11] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, and Y. Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012.
- [12] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.
- [13] V. Garnero, K. Junosza-Szaniawski, M. Liedloff, P. Montealegre, and P. Rzażewski. Fixing improper colorings of graphs. *CoRR*, abs/1607.06911, 2016.
- [14] O. Goldreich. On promise problems: A survey. In *Theoretical Computer Science*, volume 3895 of *Lecture Notes in Computer Science*, pages 254–290. Springer Berlin Heidelberg, 2006.
- [15] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.
- [16] R. Impagliazzo and R. Paturi. On the Complexity of k -SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
- [17] T. R. Jensen and B. Toft. *Graph coloring problems*, volume 39. John Wiley & Sons, 2011.
- [18] D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451, 1985.
- [19] M. Johnson, D. Kratsch, S. Kratsch, V. Patel, and D. Paulusma. Finding shortest paths between graph colourings. In *Proceedings of the 9th International Symposium on Parameterized and Exact Computation, IPEC 2014, Wroclaw, Poland, September 10-12*, pages 221–233, 2014.
- [20] K. Junosza-Szaniawski, M. Liedloff, and P. Rzażewski. Fixing improper colorings of graphs. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 266–276. Springer, 2015.
- [21] S. Khuller, R. Bhatia, and R. Pless. On local search and placement of meters in networks. *SIAM Journal on Computing*, 32(2):470–487, 2003.
- [22] J. Kratochvíl. Precoloring extension with fixed color bound. *Acta Math. Univ. Comen*, 62:139–153, 1993.
- [23] A. Krokhin and D. Marx. On the hardness of losing weight. *ACM Transactions on Algorithms*, 8(2):1–18, 2012.
- [24] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.
- [25] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, (105):41–72, 2011.
- [26] A. Mansfield. Determining the thickness of graphs is NP-hard. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 93, pages 9–23. Cambridge University Press, 1983.
- [27] D. Marx. Graph colouring problems and their applications in scheduling. *Electrical Engineering*, 48(1-2):11–16, 2004.

- [28] P. R. Östergård. On a hypercube coloring problem. *Journal of Combinatorial Theory, Series A*, 108(2):199–204, 2004.
- [29] S. Szeider. The parameterized complexity of k -flip local search for SAT and MAX SAT. *Discrete Optimization*, 8(1):139–145, 2011.
- [30] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.
- [31] D. West. Chromatic Villainy of Graphs. <http://www.math.illinois.edu/~dwest/regs/chromvil.html>, 2013 (accessed August 3, 2015).
- [32] M. Wrochna. Reconfiguration in bounded bandwidth and treedepth. *arXiv preprint arXiv:1405.0847*, 2014.